An Implementation of Scatter Search to Classify Medical Images

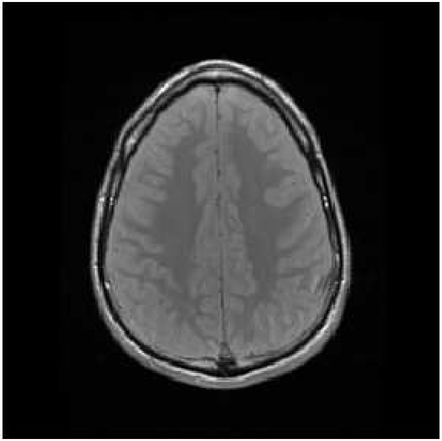
Jeffrey Larson
Jeffrey.Larson@ucdenver.edu

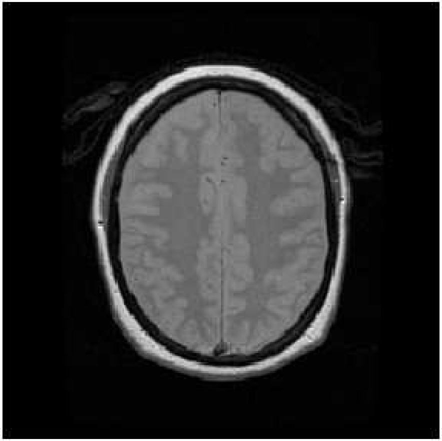
University of Colorado Denver

March 14, 2009

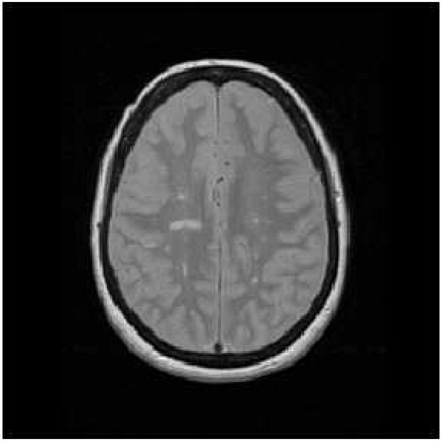
Overview

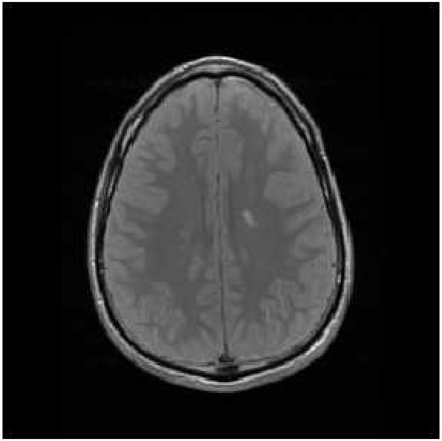
Normal brain scans



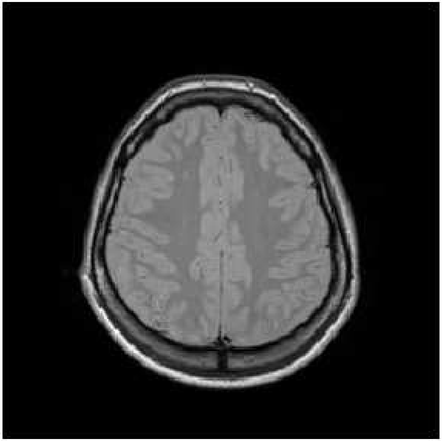


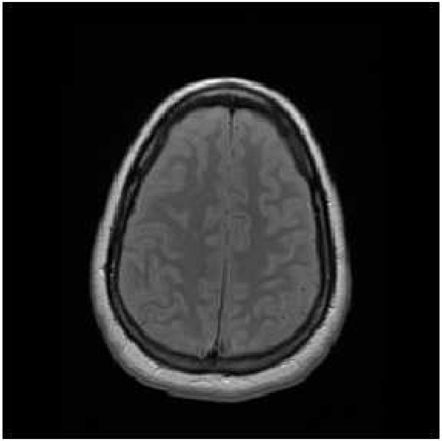
Abnormal brain scans





But it's not always easy to tell...





Feature Vectors

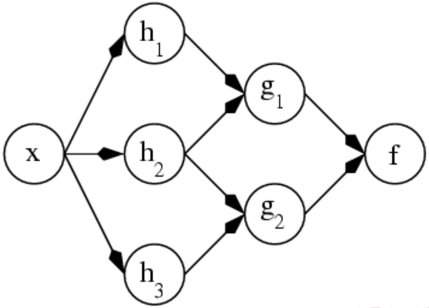
What do we do with these 256x256 matrices?

Dealing with the entire matrix is difficult
Transform the matrix:

- Spectrum of the matrix
- Coefficients from the Discrete Fourier transform
- Haralick transform
  - 14 statistics calculated from the data in the matrix.

What is a Neural Net?

- An ANN is nothing more than a large function $f(x)$ that is a composition of other functions $g_i(x)$
- The g_i's are themselves defined as a composition of functions
- The relationship between the functions can be represented as a network, with edges denoting dependencies

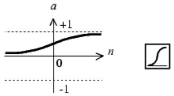


Traditional Feed-Forward ANN's

Our neural network is a function

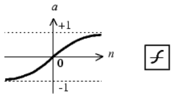
$$f(x) = F\left(\sum_i w_i g_i(x)\right)$$

where F is usually a function such as:



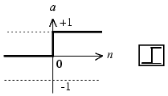
$a = logsig(n)$

Log-Sigmoid Transfer Function



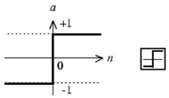
$a = tansig(n)$

Tan-Sigmoid Transfer Function



$a = hardlim(n)$

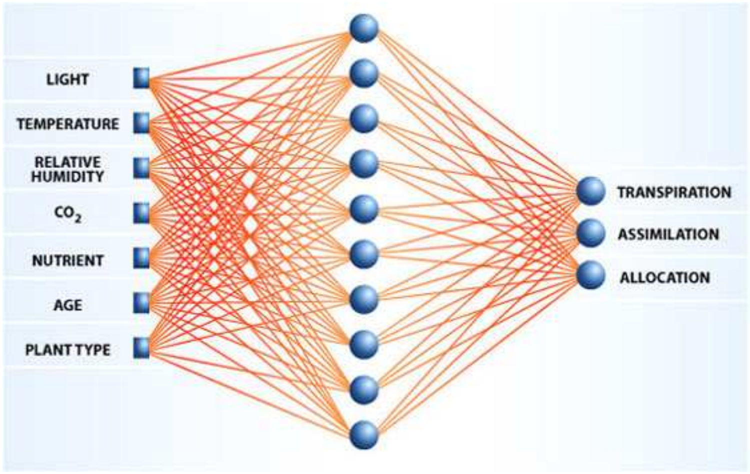
Hard-Limit Transfer Function



$a = hardlims(n)$

Symmetric Hard-Limit Transfer Function

Example from NASA



http://aemc.jpl.nasa.gov/activities/bio_regen.cfm

What is a good NN?

- Suppose we have $\{x_1, \ldots, x_N\}$ vectors and we know their outcome y_i.
- Then we'd like to minimize the difference between the network output, and the target value.

$$\min_{w} \sum_{i=1}^{N} |f(x_i) - y_i|$$

- The big question is: How can the weights be found?

Back-propagation [Werbos 1974]

- Errors from the output layer are passed back through the hidden layers
- Back-prop essentially performs steepest descent
- This is the most common way neural networks are trained, but is quite sensitive to initial conditions
- A fixed network structure must be defined beforehand

Why Scatter Search [Glover et al. 2000]

- For a fixed network architecture, training is finding the weights that best classify the training vectors
- We are trying to optimize a complex objective function with many local suboptimal minima
- Difficult problems of this style have been solved using heuristics before

General Implementation

- Scatter Search is a population-based meta heuristic with an general framework that is easily adaptable to many different problems:
  1. Generate a starting population
  2. Perform a local search on the population (if possible)
  3. Form a reference set of good solutions and diverse solutions using an appropriate metric
  4. Form appropriate subsets from elements in the reference set
  5. For each subset, generate new member(s) of the population
  6. Return to (2) and repeat until an adequate solution is found, or time runs out

A Scatter Search Implementation for ANN's

1. Generate 105 different networks with weights pulled uniformly from $[-1, 1]$
2. Find a local minimum of our error function using Nelder-Mead
3. Take the five networks with the smallest error to form the good part of the reference set. For five iterations, take the network with the furthest Euclidean distance from the reference set and add it to the reference set.
4. Generate every two element subset of the reference set
5. For each two element subset $\{x, y\}$ form three points

Training with 10 normals and 10 abnormals

Training Set	Training Error	Norms/Abs Classified	Classification Rate	Backprop Rate
First 10	1.2×10^{-8}	186/71	73%	70%
Random 10(a)	8.23×10^{-5}	199/56	77%	67%
Random 10(b)	2.48	185/48	71%	59%
Random 10(c)	0.49	209/56	80%	52%
Random 10(d)	4.32	206/53	78%	56%
Random 10(e)	3.50	210/57	81%	66%
Random 10(f)	0.93	178/83	79%	62%

Networks of different size

Training Set	Hidden Nodes	Training Error	Classification Rate
Random 10(a)	5	6.67	75%
Random 10(b)	5	7.57	67%
Random 10(c)	5	9.23	69%
Random 10(d)	5	3.64	78%
Random 10(e)	5	3.50	80%
Random 10(f)	5	10.18	42%
Random 10(a)	20	6.86	60%
Random 10(b)	20	1.36	70%
Random 10(c)	20	3.00	73%
Random 10(d)	20	3.64	77%
Random 10(e)	20	3.67	80%
Random 10(f)	20	3.50	72%

Some possible extensions:

- Further compare networks of different sizes
- Compare different feature vectors
- Change the recombination procedure, local search, etc
- Sensitivity Analysis

Thank you
Questions?

If you want any of my code, just email me:
`Jeffrey.Larson@ucdenver.edu`